

Automated Word Puzzle Generation via Topic Dictionaries

Balázs Pintér
Gyula Vörös
Zoltán Szabó
Lőrincz András

BLI@ELTE.HU
VOROSGY@INF.ELTE.HU
SZZOLI@CS.ELTE.HU
ANDRAS.LORINCZ@ELTE.HU

Faculty of Informatics, Eötvös Loránd University, Pázmány P. sétány 1/C, H-1117 Budapest, Hungary

1. Introduction

Puzzles play a central role in our everyday lives with exciting potentials. As assessments in education and psychometry, puzzles are among the most frequently used tools (Verguts & Boeck, 2000). A well-known example is the *odd one out* puzzle of IQ tests (Carter, 2005). There exist dedicated word puzzles to test or improve a wide array of skills including language skills, verbal aptitude, logical thinking or general intelligence, such as the multiple-choice synonym task of the TOEFL test. Puzzle creation is also a vibrant subfield of procedural content generation for games (PCG), the automated generation of game content for which there is a continuously increasing demand thanks to the thriving popularity of computer and video games.

However, generating and maintaining such puzzles *manually* is quite challenging and expensive: automated schemes could be of considerable benefit. A central problem one has to cope with is variety; otherwise the solver will encounter the same (kind of) puzzle multiple times. In case of word puzzles, new puzzles are needed continuously because (i) novel words are created on a daily basis (e.g., on blogs), (ii) existing words get new meaning (e.g., ‘chat’), (iii) words go out of common use (e.g., ‘videotape’). Due to the different nature of puzzles, the problem of generating puzzles automatically has been tackled only in quite special cases. There exist, for example efficient techniques for (i) sudoku games, (ii) creating mazes on chessboards, or (iii) generating puzzles and quests (objectives for the players) for massively multiplayer online games.

To the best of our knowledge, automated *word puzzle* generation is a novel area of this field. Colton (2002) addressed the problem by a complex theory formation

system to obtain *odd one out*, *analogy* and *next in the sequence* puzzles. The presented approach however relied on highly structured datasets, which required serious human annotation effort.

Our goal is to develop a general automated word puzzle generation method from

1. an unstructured and unannotated document collection, i.e., a simple *corpus*,
2. a *topic model*¹, which induces a topic dictionary from the input corpus, and
3. a *semantic similarity* measure of word pairs.

Our method, relying only on these three general components, is capable of (i) generating automatically a large number of valuable word puzzles of *many different types*, including the odd one out, choose the related word and separate the topics puzzle:

- In *odd one out* puzzles, the solver is required to select the word that is dissimilar to the other words.
- In *choose the related word* puzzles, the solver has to select the word that is closely related to a previously specified group of words.
- In *separate the topics* puzzles, the solver has to separate the set of words into two disjoint sets of related words.

(ii) The method can create easily *domain-specific* puzzles by replacing the corpus component. (iii) It is also capable of automatically generating puzzles with *parameterizable levels* suitable for, e.g., beginners or intermediate learners. In the following, we present the basic ideas behind our approach (Section 2) with some numerical illustrations (Section 3). For more extensive demonstrations and further details, see (Pintér et al., 2012).

¹Examples include e.g., latent semantic analysis, group-structured dictionaries or latent Dirichlet allocation.

2. Method

Below, we define the key components of our presented approach for automated word puzzle generation. The word puzzles we focus on are produced by (i) generating *consistent sets* of related concepts and then (ii) mixing these sets with weakly related elements: words or other consistent sets. For example, in the *odd one out* puzzle, it is sufficient to add a single unrelated word to a consistent set.

For the generation of consistent sets (see Algorithm 1), we assume that we are given (i) an *unlabeled corpus* $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$ and (ii) a *topic model* \mathcal{T} . In corpus \mathbf{X} , the documents ($\mathbf{x}_i \in \mathbb{R}^N$) are represented as weights assigned to words. For example, in a bag of words representation x_{ij} is the number of occurrences of the i th word in the j th document. Our assumption for the topic model \mathcal{T} , is that it induces a *dictionary* $\mathbf{D} = \mathcal{T}(\mathbf{X}) = [\mathbf{d}_1, \dots, \mathbf{d}_K] \in \mathbb{R}^{N \times K}$ whose K elements, i.e., *topics* \mathbf{d}_i ($i = 1, \dots, K$) describe well the documents in the corpus.

Numerous topic models (\mathcal{T}) fit to this family. For example, in latent semantic analysis (LSA; (Deerwester et al., 1990))—which is perhaps the most widely known topic model—the singular value decomposition of $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ is computed and \mathbf{X} is approximated by keeping only the first K (columns of \mathbf{U}) left singular vectors; these vectors form \mathbf{D} . Group-structured dictionaries approximate \mathbf{X} by adding a structure-inducing regularization (Ω) on the elements of \mathbf{D} , i.e., minimize the cost function ($\rho \geq 0$)

$$\min_{\mathbf{D}} \frac{1}{\sum_{j=1}^M (j/M)^\rho} \sum_{i=1}^M \left(\frac{i}{M} \right)^\rho l(\mathbf{x}_i, \mathbf{D}), \quad (1)$$

$$l(\mathbf{x}, \mathbf{D}) = \min_{\alpha} \left[\frac{1}{2} \|\mathbf{x} - \mathbf{D}\alpha\|_2^2 + \kappa \Omega(\alpha) \right] \quad (\kappa > 0). \quad (2)$$

For an excellent review on structured sparsity, see (Bach et al., 2012). In latent Dirichlet allocation (LDA; (Blei et al., 2003)) topics $\mathbf{d}_i \in \mathbb{R}^N$ ($i = 1, \dots, K$) are modelled as latent random variables with a Dirichlet prior. The dictionary \mathbf{D} consists of the estimated \mathbf{d}_i s.

For word puzzles, we keep only the k ($\leq N$) most significant words of the topics as sets: $m_i = \arg \max_k (\mathbf{d}_i) \subseteq \{1, \dots, N\}$ ($|m_i| = k$). Topic models can produce junk topics (Alsumait et al., 2009). For example, common function words, such as *did*, *said*, etc. can form a topic. These topics result in inconsistent sets, whose words are not closely related. To evaluate the consistency of sets and discard inconsistent ones—which is highly desirable in word puzzles—we define the consistency of the resulting word sets m_i

using the semantic relatedness of the word pairs they contain measured by explicit semantic analysis (ESA; (Gabrilovich & Markovitch, 2009)). In ESA, given a concept repository, such as the articles of Wikipedia, the relatedness of two words (w, w') is measured as the similarity of their concept based representations:

$$s_{ww'} = \cos(\varphi_{ESA}(w), \varphi_{ESA}(w')). \quad (3)$$

The basic assumption of ESA is that if a word appears frequently in a Wikipedia article, then that article represents the meaning of the word well.

Since in word puzzles, even a single word too weakly connected to the others can make the resulting puzzles ambiguous, it is prudent to rate each set (m_i) according to the *word that is the least related to the other words in the set*. The semantic relatedness measure (Eq. (3)) is also not perfectly accurate: false positives (two words seem related when in reality they are not) or false negatives (the similarity measure gives a small value even though the two words are related) may appear.

To cope with these challenges in determining set (m) consistency, one can proceed as follows (see Algorithm 1, line 7). Robustness to false negatives can be increased by defining the relatedness of two words based on *all paths* ($\max_{path(i,j)} s_{uv}$) between them in the semantic similarity (s_{uv} , Eq. (3)) graph $G = (m, \mathbf{S}|_m)$. Robustness to false positives can be achieved by taking the *minimum relatedness on the path* ($\min_{e \in path(i,j)} s_e$). Finally, to ensure that the quality of a set is determined by the two most dissimilar words in the set, one can compute the minima over all $i \neq j$ word pairs ($\min_{i \neq j} sim(i, j)$). A set is defined to be *consistent* if this quality of the set is above a given threshold δ . It can be shown (Jungnickel, 2007) that the $sim(i, j)$ similarity values are equal to the weight of the unique path between i and j in the maximum spanning tree (T) of G . Moreover, since the s_{uv} values are non-negative, it is sufficient to find the edge with minimal weight in T to determine the quality of the set. For an illustration, see Fig. 1.

Having the consistent sets (\mathcal{C}) at hand, word puzzles can be easily generated by mixing unrelated elements with \mathcal{C} . The pseudocode of *odd one out* puzzle generation is given in Algorithm 2. The puzzle generator has two parameters, η_1 and η_2 . Parameter η_2 determines whether the consistent set and the unrelated element are dissimilar enough so that they can be mixed to form a word puzzle. Parameter η_1 allows the creation of puzzles of different difficulty (beginner, intermediate, etc.): by increasing η_1 , the relatedness of the additional elements to the consistent set is increased, therefore, the puzzle is made harder. Similar

Algorithm 1 Identify Consistent Sets (\mathcal{C})

- 1: **Input:** corpus $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$, topic model \mathcal{T} , size of consistent sets k , semantic similarity of words $\mathbf{S} = [s_{ij}] \in \mathbb{R}^{N \times N}$, consistency threshold δ
- 2: $\mathcal{C} \leftarrow \emptyset$ // there is no consistent set at the beginning
- 3: $\mathbf{D} = \mathcal{T}(\mathbf{X}) = [\mathbf{d}_1, \dots, \mathbf{d}_K] \in \mathbb{R}^{N \times K}$ // compute the topic dictionary
- 4: $\mathcal{M} = \{m_1, \dots, m_K\}$, $m_i = \arg \max_k (\mathbf{d}_i) \subseteq \{1, \dots, N\}$, $|m_i| = k$ // k most significant words of the topics
- 5: **for all** $m \in \mathcal{M}$ **do**
- 6: $G = (m, \mathbf{S}|_m)$ // semantic-weighted graph of the candidate consistent set m
- 7: **if** $\min_{(i,j) \in m \times m, i \neq j} \text{sim}(i, j) := \max_{\text{path}(i,j)} \min_{e \in \text{path}(i,j)} s_e > \delta$ **then** // similarity of the 2 most dissimilar words
- 8: $\mathcal{C} \leftarrow \mathcal{C} \cup \{m\}$ // set m is declared to be consistent

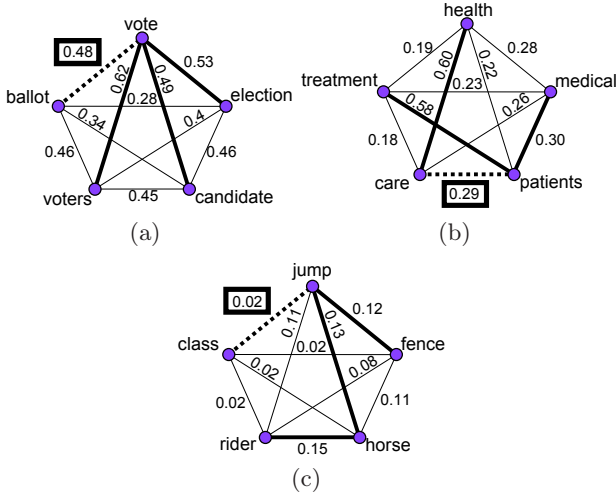


Figure 1. Checking the consistency of 3 sets of words. Here, sets contain $k = 5$ words. Bold edges: maximal spanning tree. Dashed line: edge with minimal weight in the tree; determines consistency. (a): a highly consistent set; all the words are strongly connected to the word *vote*. (b): a consistent set; some of the relatedness values (e.g., between *care* and *treatment*) are lower than one would expect; The method is robust: a relatively high consistency value is assigned to the set. (c): an inconsistent set; the word *class* is weakly connected to the others.

constructions can be applied to generate *choose the related word* or *separate the topics* puzzles (Pintér et al., 2012).

3. Illustration

Here, we illustrate the efficiency of our method in automated *odd one out* puzzle generation.

Consistent sets are a cornerstone of the presented method. In the **first experiment** we compared the number of consistent sets of a given quality ($\geq \delta$) (i) the different topic models, LSA, LDA and OSDL (Szabó et al., 2011) a recent group-structured dictio-

Algorithm 2 Odd One Out Puzzle Generation

- Input:** consistent sets \mathcal{C} , minimal (maximal) relatedness to consistent sets η_1 (η_2)
- for all** $C \in \mathcal{C}$ **do**
- repeat**
- select random word w
- $\sigma \leftarrow \max_{t \in C} s_{tw}$ // max. relatedness of w to C
- until** $\eta_1 < \sigma < \eta_2$
- output (C, w) puzzle

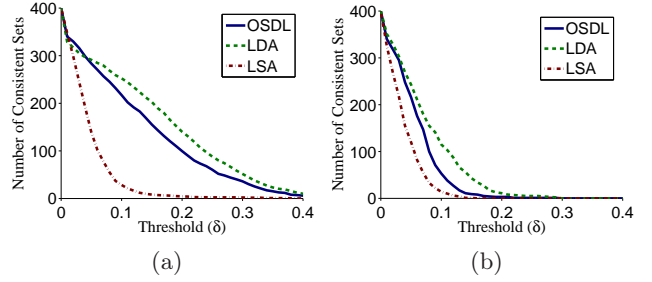


Figure 2. Number of consistent sets produced by the different topic models as a function of threshold δ . (a): corpus of Wikipedia, (b): NIPS proceedings.

nary learning technique could produce, (ii) on two corpora (\mathbf{X}). The two corpora were the English Wikipedia with $M = 10,000$ samples and the domain-specific corpus of NIPS proceedings ($M = 1,740$). Consistent sets were composed of $k = 4$ words. The number of topics was chosen to be $K = 400$. Our results are summarized in Fig. 2. According to the figure, out of the studied topic models, LDA performs the best, with OSDL following closely behind. LSA does not seem applicable to word puzzle generation: it produces very few consistent sets. The methods perform better in terms of the number of consistent sets on Wikipedia than on the corpus of NIPS proceedings. This is expected, since M , the number of articles in the Wikipedia highly exceeded that of NIPS proceedings.

Consistent set of words				Odd one out
vote	election	candidate	voters	sony
church	orthodox	presbyterian	evangelical	buddhist
olympic	tournament	world	championship	acid
austria	german	austrian	vienna	scotland
devil	demon	hell	soul	boat
harry	potter	wizard	ron	manchester
superman	clark	luthor	kryptonite	division
magic	world	dark	creatures	microsoft

Table 1. Odd one out – beginner puzzles.

Consistent set of words				Odd one out
cao	wei	liu	emperor	king
superman	clark	luthor	kryptonite	batman
devil	demon	hell	soul	body
egypt	egyptian	alexandria	pharaoh	bishop
singh	guru	sikh	saini	delhi
language	dialect	linguistic	spoken	sound
mass	force	motion	velocity	orbit
voice	speech	hearing	sound	view
athens	athenian	pericles	corinth	ancient
function	problems	polynomial	equation	physical

Table 2. Odd one out – intermediate puzzles.

In the **second experiment**, we demonstrate the obtained *odd one out* puzzles generated from Wikipedia (X). We chose $\delta = 0.1$ to obtain a significant number of good enough consistent sets (see Fig. 2). First, we illustrate the beginner puzzles (Table 1), where we chose parameters $\eta_1 = 0.005$, and $\eta_2 = 0.02$. In other words, the puzzles generated consist of a consistent set of related words and an unrelated word. Beginner puzzles can be solved at first glance by a person who understands the language and has a wide vocabulary, for example, the puzzles *vote*, *election*, *candidate*, *voters*, *sony*, or *olympic*, *tournament*, *world*, *championship*, *acid*. These could be useful for e.g., beginner language learners or, with a suitable corpus, for children. Some puzzles require specific knowledge about a topic. To solve the puzzles *harry*, *potter*, *wizard*, *ron*, *manchester* and *superman*, *clark*, *luthor*, *kryptonite*, *division*, the solver must be familiar with the book, film, comic, etc. To solve *austria*, *german*, *austrian*, *vienna*, *scotland*, geographic knowledge is needed.

Second, we illustrate intermediate puzzles (Table 2) obtained with $\eta_1 = 0.1$ and $\eta_2 = 0.2$. Although the presented method is based on semantic similarity, it is able to create surprisingly subtle puzzles. In the puzzle *voice*, *speech*, *hearing*, *sound*, *view*, the word *view* has a different modality than the others. To solve the puzzle *cao*, *wei*, *liu*, *emperor*, *king*, the solver should be familiar with the three kingdoms period of the Chi-

nese history. For *egypt*, *egyptian*, *alexandria*, *pharaoh*, *bishop*, knowledge of the Egyptian history, for *athens*, *athenian*, *pericles*, *corinth*, *ancient*, familiarity with the Peloponnesian War is required. In *singh*, *guru*, *sikh*, *saini*, *delhi*, all the words except *delhi* are related to sikhism. The puzzle *function*, *problems*, *polynomial*, *equation*, *physical* can be solved only with a basic knowledge of mathematics and physics. These results demonstrate the efficiency of our automated word puzzle generation approach.

Acknowledgments. The European Union and the European Social Fund have provided financial support to the project under the grant agreement no. TÁMOP 4.2.1./B-09/1/KMR-2010-0003. The research has also been supported by the ‘European Robotic Surgery’ EC FP7 grant (no.: 288233). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of other members of the consortium or the European Commission.

References

- Alsumait, L., Barbará, D., Gentle, J., and Domeniconi, C. Topic significance ranking of LDA generative models. In *ECML PKDD '09*, pp. 67–82, 2009.
- Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. Optimization with sparsity-inducing penalties. *Found. Trend. Mach. Learn.*, 4(1):1–106, 2012.
- Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.
- Carter, P. *IQ and Psychometric Test Workbook*. Kogan Page Business Books, 2005.
- Colton, S. Automated puzzle generation. In *AISB'02, Imperial College, London*, 2002.
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *J. Am. Soc. Inform. Sci.*, 41(6):391–407, 1990.
- Gabrilovich, E. and Markovitch, S. Wikipedia-based semantic interpretation for natural language processing. *J. Artif. Intell. Res.*, 34:443–498, 2009.
- Jungnickel, D. *Graphs, Networks and Algorithms*. Springer; 3rd edition, 2007.
- Pintér, B., Vörös, Gy., Szabó, Z., and Lőrincz, A. Automated word puzzle generation using topic models and semantic relatedness measures. In *Annales Univ. Sci. Budapest., Sect. Comp.*, volume 36, pp. 299–322, 2012. http://ac.inf.elte.hu/Vol_036_2012/299_36.pdf.
- Szabó, Z., Póczos, B., and Lőrincz, A. Online group-structured dictionary learning. In *CVPR*, pp. 2865–2872, 2011.
- Verguts, T. and Boeck, P. De. A Rasch model for detecting learning while solving an intelligence test. *Appl. Psych. Meas.*, 24(2):151–162, 2000.